Hangman Game

Assignment 3

2020-03-1

Xiaohe Zhu

xz222az@student.lnu.se

Iteration 3

Time: 2020-02-25 to 2020-03-9

Plan and Time Log

Task	Start date	Estimate time	Actual time	Difference
View video	3-3	3 h	2.5 h	0.5 h
Time log	3-4	10 min	10 min	0
Test plan	3-4	30 min	45 min	15 min
Manual test	3-4	3 h	2.5 h	0.5 h
Unit test	3-5	2 h	2 h	0
My reflection	3-8	15 min	15 min	0

Objective

The objective of this project is to test the code that was implemented for iteration 2 to make sure that the final project is good enough to be used.

What to Test and How

For this iteration, there are only two use cases to test which is UC1 start game and UC2 play game by writing and running dynamic manual test-case. The main reason we only chose these two use cases to be tested is the time limitation, and these use cases are essential to the project. For other use cases, we choose to write the tests in the later iteration.

Two methods from GameControl class and one unfinished method in HighScore class shall be tested by automated tests.

```
public boolean guessed(ArrayList<String> guess, String st)
public boolean isLetter(String st)
public int[] getTop5(ArrayList<Integer> history)
```

There shall be at least two unit tests per method. Due to lack of time, we chose these two methods since they have simple input that can be easily controlled by the tester, and simple output that can be easily asserted by JUnit test.

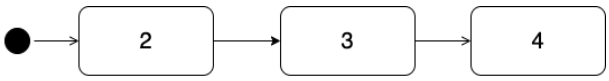
we only use dynamic tests (i.e. tests which must run through execute a program), not static test. The tests include manual tests and automated tests. The dynamic manual tests are bases on the use cases we defined in iteration 2. And for the automated test, we are using Jnit5.

Manual Test-Case

TC1.1 Start game successfully

Use case: UC1 Start Game

Scenario: Start game successfully



Description: The main scenario of UC1 is tested where a user starts game successfully (start of UC2).

Precondition: delete all words in WordsDatabase class except wave.

Test steps:

1. Start the game
2. The system shows the main menu "Do you want to play? (Enter play or quit): "
3. User types play and press enter.

Expected:

1. The system should show the game menu. (“Welcome”, “_ _ _ _”, “Guess a letter:”)

Test result:

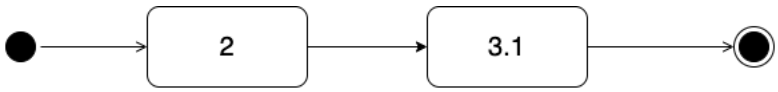
✔ Pass
☐ Fail

Comment: TC1.1 passed and enter to UC 2 which will test below in TC2.

UC1.2 Quit game

Use case: UC1 Start Game

Scenario: Quit the game



Description: The alternative scenario of UC1 is tested where a user quite game (start of UC3).

Precondition: None.

Test steps:

1. Start the game

2. The system shows the main menu "Do you want to play? (Enter play or quit): "
3. User types quit and press enter.

Expected:

1. The system shows "Are you sure to quit? (Enter quit or not)"

Test result:

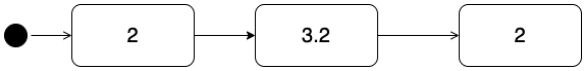
✓ Pass
☐ Fail

Comment: TC1.2 passed and enter to UC3 quit game which is not tested in this iteration.

TC1.3 Illegal menu choice and promote input again

Use case: UC1 Start Game

Scenario: User enters an illegal input



Description: The alternative scenario of UC1 is tested where a user enters an illegal input.

Precondition: None.

Test steps:

1. Start the game
2. The system shows the main menu "Do you want to play? (Enter play or quit): "
3. Type p and press enter, System shows “invalid input. Do you want to play? (Enter play or quit)”.
4. Type 1 and press enter.

Expected:

1. The system shows "invalid input. Do you want to play? (Enter play or quit)".

```
Invalid input
Do you want to play? (Enter play or quit):
```

Test result:

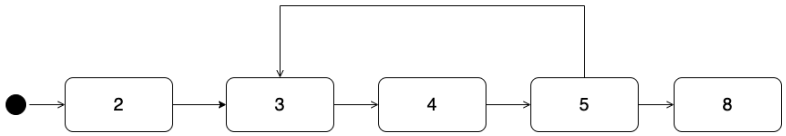
✓ Pass
☐ Fail

Comment: none.

TC2.1 Guess word successfully without incorrect guess

Use case: UC2 Play Game

Scenario: User guess the entire word without incorrect guess.



Description: The main scenario of UC2 is tested where a user guesses the word successfully.

Precondition: the game is started, and the tester deletes all words in WordsDatabase class except wave.

Test steps:

1. Start the game
2. The system shows the main menu guess menu with "_ _ _ _" and "Guess a letter".
3. Type "w" and press enter, the system shows "Correct guess", "w _ _ _ "," Guess a letter".
4. Type "a" and press enter, system shows "Correct guess", "w a _ _ "," Guess a letter".
5. Type "v" and press enter, the system shows "Correct guess", "w a v _ "," Guess a letter".
6. Type "e" and press enter.

Expected:

1. system shows "Correct guess", "w a v e "," Guess a letter", "You win and your point is 4".

Test result:

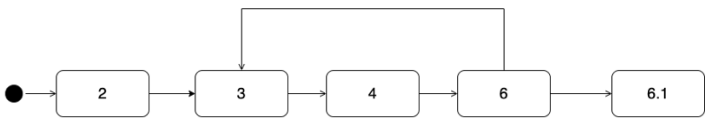
✓ Pass
☐ Fail

Comment: TC2.1 passed and enter to UC5 view high score list which is not tested in this iteration.

TC2.1 Guess word incorrectly without a correct guess

Use case: UC2 Play Game

Scenario: User guess word without correct guess.



Description: The alternative scenario of UC2 is tested where a user guesses word without a correct guess.

Precondition: the game is started, and the tester deletes all words in WordsDatabase class except wave.

Test steps:

1. Start the game
2. The system shows the main menu guess menu with "_ _ _ _" and "Guess a letter".
3. Type q and press enter, system shows “Wrong guess and you have 7 times only”, one stroke of hangman, “_ _ _ _”, “Guess a letter.

```
Wrong guess and you have 7 times only
    ________
    |
    |
    |
    |
    |
========


_ _ _ _
Guess a letter:
```

4. Type r and press enter, system shows “Wrong guess and you have 6 times only”, one stroke of hangman, “_ _ _ _”, “Guess a letter.

```
Wrong guess and you have 6 times only
    ________
    |    |
    |
    |
    |
    |
========


_ _ _ _
Guess a letter:
```

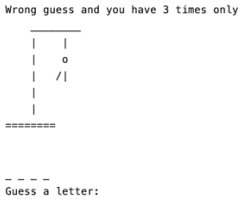
5. Type t and press enter, system shows “Wrong guess and you have 5 times only”, one stroke of hangman, “_ _ _ _”, “Guess a letter.

```
Wrong guess and you have 5 times only
    ________
    |    |
    |    o
    |
    |
    |
========


_ _ _ _
Guess a letter:
```

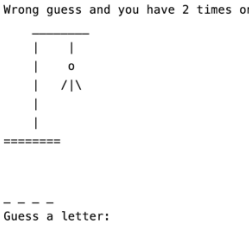
6. Type y and press enter, system shows “Wrong guess and you have 4 times only”, one stroke of hangman, “_ _ _ _”, “Guess a letter.

```
Wrong guess and you have 4 times only
    ________
    |    |
    |    o
    |    |
    |
    |
========


_ _ _ _
Guess a letter:
```

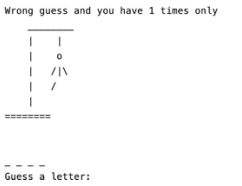
7. Type u and press enter, system shows “Wrong guess and you have 3 times only”, one stroke of hangman, “_ _ _ _”, “Guess a letter.



8. Type g and press enter, system shows “Wrong guess and you have 2 times only”, one stroke of hangman, “_ _ _ _”, “Guess a letter.



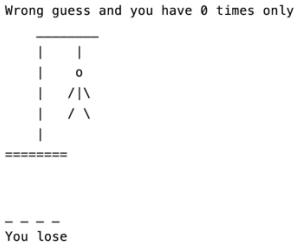
9. Type h and press enter, system shows “Wrong guess and you have 1 times only”, one stroke of hangman, “_ _ _ _”, “Guess a letter.



10. Type r and press enter.

Expected:

1. the system shows "Wrong guess and you have 0 times only”, one stroke of hangman, “_ _ _ _”, “You lose”.



Test result:

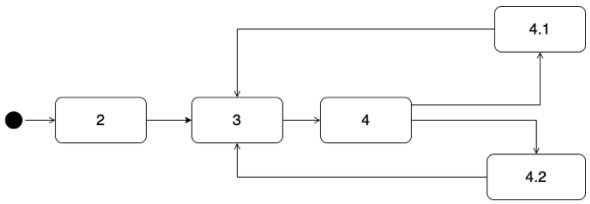
✔ Pass
☐ Fail

Comment: TC2.1 passed and enter to UC5 view high score list which is not tested in this iteration

TC2.3 Illegal guess and promote input again

Use case: UC2 Play game

Scenario: User enters an illegal input



Description: The alternative scenario of UC2 is tested where a user enters an illegal input.

Precondition: the game is started, and the tester deletes all words in WordsDatabase class except wave.

Test steps:

1. Start the game
2. The system shows the main menu guess menu with "_ _ _ _" and "Guess a letter".
3. Type "w" and press enter, the system shows "Correct guess", "w _ _ _", “Guess a letter”.
4. Type “1" and press enter, the system shows "Invalid input”, “w _ _ _”, “Guess a letter”.
5. Type “wave" and press enter, the system shows "Invalid input”, “w _ _ _”, “Guess a letter”.
6. Type “w” and press enter.

Expected:

1. The system shows "You have guessed this letter before”, “w _ _ _”, “Guess a letter”.”.

Test result:

✔ Pass
☐ Fail

Comment: none.

Test Report

Test	UC1	UC2	UC3	UC4	UC5
TC1.1	1/OK	0	0	0	0
TC1.2	1/OK	0	0	0	0
TC1.3	1/OK	0	0	0	0
TC2.1	0	1/OK	0	0	0
TC2.2	0	1/OK	0	0	0
TC2.3	0	1/OK	0	0	

COVERAGE & SUCCESS	3/OK	3/OK	0	0	0

Comment:

All tests pass, UC3, UC4, and UC5 will be tested on the next iteration due to the time limitation.

Automated Unit Test

Two methods from GameControl class and one unfinished method in HighScore class shall be tested by automated tests.

`public boolean guessed(ArrayList<String> guess, String st)`

`test code`

```
import ...

class GameControlTest {
    private ArrayList<String> strings = new ArrayList<>();
    GameControl gameControl = new GameControl();
    private ArrayList<Integer> history = new ArrayList<>();
    HighScore highScore = new HighScore();


    @Test
    //when guess the first letter, the arrayList is empty
    void guessed1() {
        boolean actual = gameControl.guessed(strings, st: "GuessOne");
        assertFalse(actual);
    }

    @Test
    //when one guess already be made, the arrayList is not empty, the arrayList contain the guess string
    void guessed2() {
        strings.add("Guessed");
        boolean actual = gameControl.guessed(strings, st: "Guessed");
        assertTrue(actual);
    }

    @Test
    //when one guess already be made, the arrayList is not empty the arrayList dosen't contain the guess string
    void guessed3() {
        boolean actual = gameControl.guessed(strings, st: "Guess");
        assertFalse(actual);
    }
```

`public boolean isLetter(String st)`
`test code`

```java
@Test
void isLetter1(){
    String s = "word";
    boolean actual = gameControl.isLetter(s);
    assertFalse(actual);
}
@Test
void isLetter2(){
    String s = "w";
    boolean actual = gameControl.isLetter(s);
    assertTrue(actual);
}
@Test
void isLetter3(){
    String s = "1";
    boolean actual = gameControl.isLetter(s);
    assertFalse(actual);
}
```

public int[] getTop5(ArrayList<Integer> history)

test code

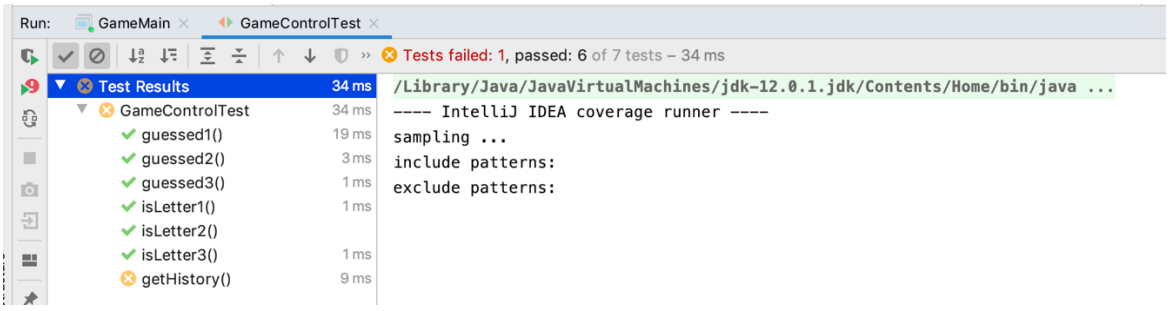
```java
@Test
//test the method in HighScore class
void getHistory(){
    history.add(3);
    history.add(1);
    history.add(5);

    int[] top5 = new int[5];
    top5 = highScore.getTop5(history);

    //the top5 should be sorted
    //implement the comment in HighScore class will let the test success
    for (int i = 0;i<4;i++) {
        assertTrue( condition: top5[i] >= top5[i+1]);
    }
}
}
```

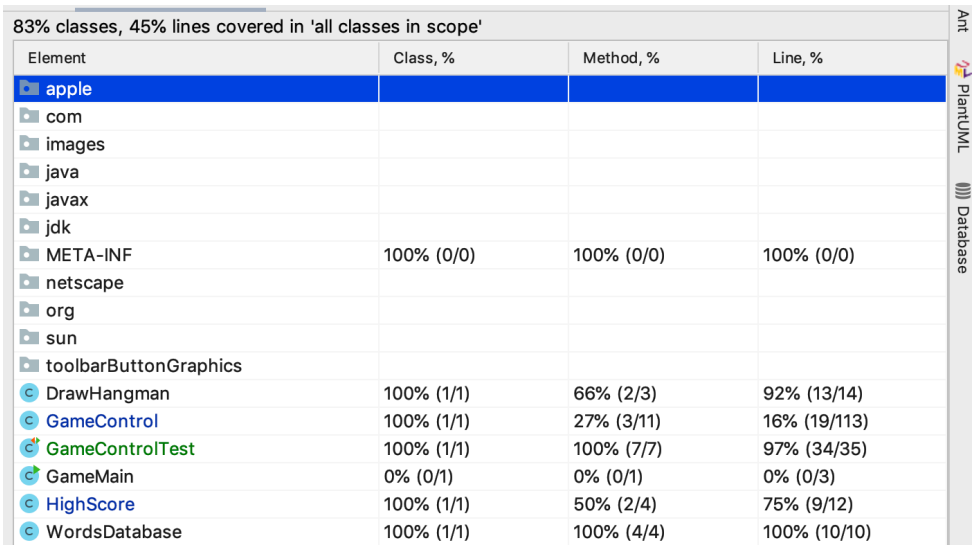
Entire source code can be found:

https://gitlab.lnu.se/1dv600/student/xz222az/assignment-2/tree/master/1DV600code

Test result

6 tests pass and one unfinished method fail which will success when removing the comment in HighScore class





My reflection

Due to lack of time, I only create two test cases based on two use cases that defined on iteration 2, and many use cases remain untested now. And for writing manual tests and trying to find every path for each use case, I made a little improvement in the definition of use cases.

The same problem happens to automated unit tests too. I only cover two finished methods and one unfinished method. One reason is lack of time and another main reason is that all of my method depends on other methods or class, and lots of my methods are void methods that do not have clear input and output, which makes it hard to be the test. To improve the testability, I need to make a lot of changes in my source code.